IMPLEMENTATION OF MS - WINDOWS BASED SNMP MANAGER

by

Bhaskar Bhar

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

March, 1996

IMPLEMENTATION OF MS-WINDOWS BASED SNMP MANAGER

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Bhaskar Bhar

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

March 1996

Certificate

It is certified that the work contained in the thesis entitled **Implementation of MS-Windows based SNMP manager,** by **Bhaskar Bhar**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

K R Srivathsan

K. R. Srivathsan
Professor
Department of Electrical Engineering
IIT, Kanpur.

March 1996

Abstract

Management of large networks, consisting of many interconnected network segments, is done by collecting traffic statistics, gathering routing information, reporting errors, monitoring link equipments, servers, systems, and locating faults. A management station is used to collect the nessary information from the different systems on the network and display suitably relevant processed information. The communication between the manager and managed systems uses special network management protocols. SNMP (Simple Network Management Protocol) is currently the most popular network management protocol for TCP/IP based networks. SNMP was originally targeted at the TCP/IP based networks, but its rapid adoption and easy extendibility have caused its use to spread into proprietary environments.

Earlier implementations of SNMP in the "ERNET" Lab were PC based SNMP manager with basic functionalities by Barari [barari93] and MicroVax (Ultrix) based SNMP manager with user interface by Bansal [bansal94]. A proxy agent was implemented on the same machine by Bansal to poll machines without an SNMP agent. In this thesis, an MS-Windows based SNMP manager has been successfully implemented. The manager uses 'winsock' library functions for communication with agents and proxy agent(s). This network manager displays the status of the network by probing the different hosts via remote proxy agent(s) and can directly fetch MIB values from hosts with SNMP agents. In this thesis, the manager has been implemented with an added feature of polling remote proxy agent running on Linux environment. This enables the manager to distribute the polling schedules to proxy agents on systems located in different segments, thereby significantly reducing intersegment management traffic.

Contents

List of Figures

Chapter 1

INTRODUCTION

In today's world, each major organisation be it a commercial firm or educational institution has it's own local area network to support computer to computer communications. Also, interconnection between these networks is not a privilege but a necessity as minutes do make a difference today. With this internetworking and fast communication ruling our lives and especially business world, a failure of link or a bottleneck for traffic can play havoc with the network. A few business deals and some important emails delivered late can be potential loss to many. In this context, Network Management comes into play which continually apprises the authority about the health of the network.

1.1 Internetworking

Before proceeding further let us understand internetworking and internet. In simple, internetworking is the technology that makes it possible to interconnect many disparate physical networks. It hides the low-level details and makes the collection of networks appear like a single large network. Such an interconnection scheme is called an internetwork or *internet.*

An internet generally consists of a large number of systems such as PCs, mainframes, workstations, network systems etc interconnected through gateways, routers, etc. One of the primary aim of internetworking is network independence. That is, the set of operations used to establish communication or to transfer data should remain independent of the underlying network technologies and destination machine. The largest internetworking technology is the Internet Suite of Protocols, commonly referred to as TCP/IP.

1.2 Network Management

Network Management as a term has many definitions depending on whose operational function is in question. In the context of the internet, Network Management may be thought of as facilities for updating of routing information, traffic statistics collection, error reports and fault location facilities, accounting for the use of costly communication resources and network access control.

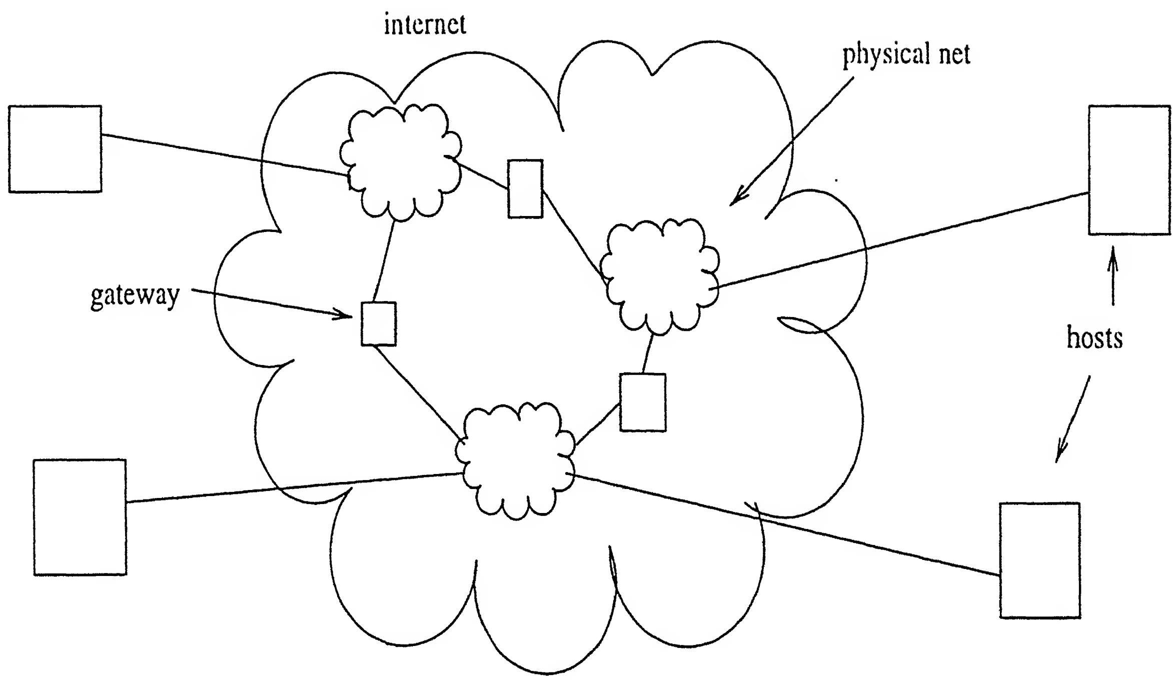

Figure 1.1: Internetworking

Proprietary Network management systems such as Netview, AT&T Accumaster and Digital Equipment Corporation's DMA, have been in operation for many years.

The International Standards Organisation has defined five key areas [dallas88] of network management. These are fault management, configuration management, security management, performance management, and accounting management. It can formally be defined as the process for controlling a complex data network so as to maximize its efficiency and productivity. The best way to manage a network is to cover each of the above five areas efficiently.

The primary goal of network management is to provide the authority with details about the health of the network so that performance of the network does not suffer. Another goal is to convey information regarding faults like link outages, if any. To aid in this task, network protocols are used so that the process of network management is automated (ie. run by computers) as much as possible.

There are many protocols available. The two mainstream protocols are SNMP (Simple Network Management Protocol) and CMIP (Common Management Information Protocol). Generally, SNMP works under the TCP/IP (Transport Control Protocol/ Internet Protocol) communication stack and CMIP works under the OSI (Open Systems Interconnection) communication stack.

1.3 History of Internet Management

Initial effort towards this started with a straightforward protocol termed as Simple Gateway Monitoring Protocol (SGMP). It received quite a good response. A second

effort began as a research project which was termed as High-level Entity Management Systems (HEMS). Although this had some novel concepts, never saw the light of day outside its development sites.

CMIP (Common Management Information Protocol) [rose91] was recommended and standardized by ISO (International Organization for standardization) for networks based on the OSI (Open System Interconnection) suite of protocols. CMIP was interfaced with TCP/IP to manage TCP/IP based networks and this was termed as CMIP over TCP (CMOT). CMIP and CMOT were complex protocols[1], hence CMOT did not became popular for TCP/IP based networks. SNMP (Simple Network Management Protocol) was released as successor of SGMP for TCP/IP based internets. CMIP remained as the management protocol for OSI based internets. CMIP is described briefly in sec 2.4.3.

In 1993 SNMP version 2 was released which addressed many shortcomings of the first version like privacy of data and authentication of user. A brief overview of this is given in sec 2.4.2. In contrast to centralized management, distributed management framework has also been proposed which seems to be the answer to management of large sized networks [INM4].

Some of the teams that are currently working in this field are Columbia University (DCC Lab), Munich Network Management Team, University of Geneva (Management of Distributed Applications using DMT), TU Delft (Data Network Performance analysis Project at Delft University of Technology), TU Wien (A concept and implementation of Hierarchical Distributed Network Management)

1.4 Objective of the Thesis

The implementation of SNMP was first tried in IIT, Kanpur by Bapat [bapat92] in his M. Tech thesis work. He studied the ISODE code of SNMP and configured it for potential use on the campus network. Next was Barari [barari93] who in his work, developed a simple SNMP manager with basic functionalities. Bansal [bansal94] came up with a full fledged SNMP manager and developed a proxy agent for managing hosts without SNMP agents. He also developed a user interface on a vt100 terminal. T.S.Rao has developed an SNMP manager for wide area networks.

The user interface developed by Bansal was limited in its capabilities as it was implemented using "curses" library functions, hence it was proposed to enhance the user interface using MS-Windows and transfer the manager to a less expensive system. Thus the objectives of this thesis were to cover above proposals which can be summarized as below.

(1) porting of Bansal's work to Linux environment,
(2) implementation of SNMP manager in MS-Windows environment,
(3) providing a window based man-machine interface for displaying the status of the network, and
(4) querying the MIB values from SNMP agents.

[1] http://www.undergrad.math.uwaterloo.ca/~tkvallil/snmp.html

Some of the terms used above will be explained in later chapters along with the context in which they are used.

1.5 Organisation of the Thesis

Chapter 1 introduces the network management and discusses the objective. Chapter 2 provides a brief description of the management issues and protocols. Chapter 3 gives a brief overview of the Management Framework used in implementation. Chapter 4 discusses in detail the implementation of the Network Manager and Proxy agent. Chapter 5 concludes the work and discusses the future aspects.

Appendices are provided in the end for quick references. A glossary of some important terms is also provided.

Chapter 2

MANAGEMENT ISSUES AND PROTOCOLS

There are several issues and aspects pertaining to network management like what information a protocol must convey, whether a network management protocol is to be built upon existing layers or is to be independent of it, etc. Some of these issues are discussed in this chapter. Various kinds of techniques were tried out to somehow manage networks in view of these issues. These, in the course of time, paved the way for more documented and standardized management protocols. In this chapter a brief overview of some of these protocols and techniques is presented.

2.1 Management Issues

The issues which need to be addressed in network management can be broadly classified into the following three categories

Functional Issues

This concerns the definition of management functions. The information manipulated by a protocol can be diveded into three classes [dallas88]. First is protocol information, encompassing all variable related to the operation of the protocol like snmp objects in MIB-2 [McClo91]. Second is report information which is related to the history of the entity like traffic statistics, system up time, etc. Third is environment information which describes the particular context of the protocol entity like addresses, routing tables, etc.

Integration Issues

This specifies that management functions should be an integral part of the protocol layering [fiet95, dallas88]. That is they use the protocol suite operating in the system. For example SNMP ,which manages TCP/IP networks, uses UDP/IP which forms a part of TCP/IP protocol suite itself.

Protocol Issues

This gives the rules for interaction between management entities. This is similar to the rules for interaction between peer layers of a protocol on different machines [dallas88]. For example when using SNMP (sec. 3.3), the message has to use certain variables and is to be coded using ASN.1 (Abstract Syntax Notation One) encoding [barari93].

2.2 Management Functions

The network management has to achieve some objectives regarding its functionality. A management protocol should be able to provide these. The five complementary aspects of the management [rose91, dallas88] are

Configuration Management which is the set of functions that identify, exercise control over, collect data from and provide data to manage resources for the purpose of supporting continuous operation of communication services. Dynamic updating of the configuration needs to be accomplished periodically to ensure the configuration is known.

Performance Management which is the set of functions that control and analyze throughput and error rate of the network. Performance is a key concern to most MIS (management of information system) support people. Although it is high on the list, it is considered difficult to be factual about some LAN performance issues.

Fault Management which is the set of functions responsible for detecting, isolating, and controlling abnormal network behaviour such as excessive line outages. Most systems poll the managed objects, search for error conditions and illustrate the problem in either a graphic format or a textual message. Fault management deals most commonly with events and traps as they occur on the network.

Accounting Management which is the set of functions responsible for collecting and processing data related to resource consumption in the network.

Security Management which is responsible for controlling access to network resources through the use of authentication techniques and authorization policies. However, most network management applications address only security applicable to network hardware such as someone logging into a router or bridge.

2.3 Management Techniques

In this section, some of the management techniques [fiet95, rose91] which are used, are described briefly. These techniques range from simple 'sending an echo message' to the more complex management by delegation paradigm.

2.3.1 Ad Hoc Management

Ad Hoc Management is based on the fact that it is possible to troubleshoot connectivity problems in a network with the help of functionalities provided by ICMP (Internet Control Message Packet) [see Sec 3.6]. These mechanisms are the lowest common denominators available. Ping and Traceroute are two such mechanisms.

The packet internet groper (ping) sends an ICMP echo request packet to an IP address and waits for reply. As implementation of ICMP is mandatory for network devices supporting TCP/IP and it provides a crude means of testing connectivity. Although it gives us round trip time and percentage loss, it cannot report on the general health of the network.

Traceroute sends a series of probe packets using UDP to an IP address. These are sent with monotonically increasing values of time to live (TTL). For each TTL value, a fixed number of packets are sent. This process continues until an ICMP port unreachable packet is received or some threshold is reached. Its elegance is lost as it is not entirely a foolproof operation.

2.3.2 Passive Management

In case of Ad Hoc Management, traffic (packets) is introduced into the network. However management can simply be done by examining the traffic generated by the transport protocol which is being managed. This is done by a device called packet monitor [fiet95]. This approach is called passive management and it relies on capture and analysis. However, this technique lacks depth and does not give enough information.

2.3.3 Centralized and Distributed Management

Current management systems pursue a platform-centred paradigm, where agents monitor the system and collect data, which can be accessed by applications via management protocols. This centralized paradigm can be contrasted with a decentralized paradigm [INM4] in which some or all intelligence and control is distributed among the network entities.

Centralized SNMP management (see sec 3.3)
The centralized SNMP paradigm evolved for several reasons. First, the most essential functions of network management are well related in this paradigm. Agents are not capable of performing self-management when global knowledge is required. Second, all network entities need to be managed through a common interface. Unfortunately, in many cases this strategy does not allow for data to be processed where and when it is most efficient to do so. The rapid growth in the size of networks has also brought into question the scalability of any centralized model.

Decentralized Management by Delegation
Management by Delegation (MBD) utilizes a decentralized paradigm that takes advantage of the increased computational power in network agents and decreases pressure on centralized Network Management Stations (NMS) (sec 3.2) and network bandwidth. MBD supports both temporal distribution and spatial distribution. In this paradigm

agents that are capable of performing sophisticated management functions locally can take computing pressure off of centralized NMSs, and reduce the network overhead of management messages. In MBD, instead of moving data from the agent to the NMS, where it is processed by applications, the application is moved to the agents where they are delegated to a process. Thus, management responsibilities can be shifted to the devices themselves when needed.

Decentralized management makes sense for those types of management applications that require or can take advantage of spatial distribution. Decentralization also allows one to more effectively monitor a network as performance changes over the time.

2.4 Management Protocols

As stated earlier, there are two major protocols in this field. They are SNMP for TCP/IP based networks and CMIP for OSI based networks. These major protocols' features are explained briefly.

2.4.1 SNMP

SNMP was designed in mid 1980's as an answer to the communication problems between different types of networks. It is a second generation protocol as it is developed from SGMP.

The way it works is very simple. It exchanges network information through messages, called PDU's (Protocol Data Unit). From a high-level perspective, the message (PDU) can be looked as an object that contains variables that have both titles and values. It is explained in detail in Chapter 3.

2.4.2 SNMPv2

Though popular, SNMP[1] is not a perfect network management protocol. The first deficiency is that it has large security gaps that can give network intruders access to the information carried along the network.

The latest version of SNMP, called SNMPv2 has added some security mechanisms that help combat three largest security problems - privacy of the data, authentication of user and access control. Also, manager to manager communication tools are available under SNMPv2 making it more suitable for large networks. Two new PDU's are added that are used to manipulate tabled objects.

2.4.3 CMIP

CMIP[2] was designed to build on SNMP by making up for SNMP's shortcomings and becoming a bigger, more detailed network manager. Its basic design is similar to

[1] http://web1.digital.net/~snmx/SNMX_SNMP_overview.html

[2] http://www.undergrad.math.uwaterloo.ca/~tkvalil/work.htm

SNMP, whereby PDU's are employed as variables to monitor a network. CMIP however contains 11 types of PDU's (compared to SNMP's five).

In CMIP, the variables are seen as very complex and sophisticated data structures, with many properties. These include :

1) Variable attributes: which represent the variables characteristics (its data type, whether it is writable).

2) Variable behaviours: what actions of that variable can be triggered.

3) Notifications: the variable generates an event report whenever a specified event occurs (eg. a terminal shutdown would cause a variable notification event.

As a comparison, SNMP employs only variable properties one and three from above.

The biggest feature of the CMIP protocol is that its variables not only relay information to and from the terminal (as in SNMP), but they can also be used to perform tasks that would be impossible under SNMP. For instance, if a terminal on a network cannot reach its fileserver in a predetermined amount of time, Then CMIP can notify the appropriate personnel of the event. Another advantage to the CMIP approach is that it addresses many of the shortcomings of SNMP. For instance, it has built in security management devices that support authorization, access control, and security logs.

One of the reasons why CMIP has not been commercially popular is due to the fact that it runs under the Open Systems Interconnection (OSI) network communication protocol. OSI performs almost all of the communications functions that TCP/IP does plus many more. It is thus thought to be a much more complete network communication package. However, with this increased completeness comes a massive increase in system resources that OSI takes to implement.

2.5 Summary

Management issues which influence the management protocol as well as architecture can be broadly classified as functional issues, integration issues, and protocol issues. The main functions of network management are configuration management, performance management, fault management, account management and security management. The management techniques are used so as to perform these functions effectively keeping in view, the issues. These range from Ad Hoc management to decentralized Management by Delegation. The two major protocols, which are used, are SNMP and CMIP. A short description of them were provided and their respective advantages and disadvantages summarized. In the next chapter, the management framework used in the implementation is described along with the architectural model and protocol.

Chapter 3

MANAGEMENT FRAMEWORK

3.1 Introduction

A management framework is defined by the architectural model and the set of protocols used. For example, in this implementation, the management model consists of the PC (on which the SNMP manager is implemented), several UNIX machines (acting as hosts) and a Linux machine (acting as a proxy agent). In this chapter, a description of management model and protocol used in the implementation is given. The description of SNMP agent, manager and proxy agent is also provided.

3.2 Management Architectural Model

A simple management architectural model [rose91] is a collection of managed nodes with an agent, managed nodes without an agent and one or more network management stations. A managed node may be a host system, gateway etc. The fundamental axiom which influences the design of the management systems is "*The impact of adding network management to managed nodes must be minimal, reflecting a lowest common denominator.*"

3.2.1 Network Management Station(NMS)

A network management station is a host system which is running a network management protocol (sec. 3.3) and a network management application. A network management application is developed in this thesis work and is explained in Chapter 4.

The network management station is responsible for management functions like traffic management, fault location, etc. Moreover, it runs management applications so that it is able to perform these management functions effectively and provides a user interface to the authority to help get a overall picture of the network performance.

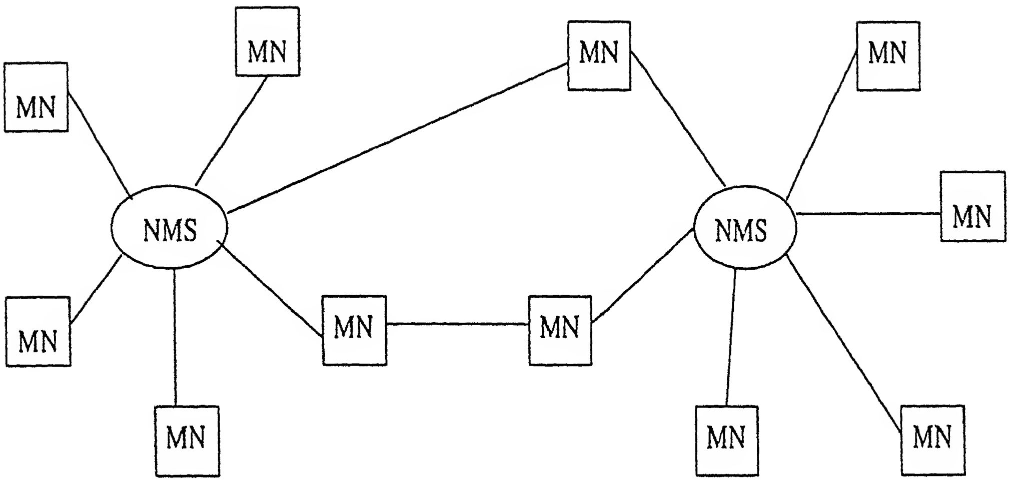

Figure 3.1: Management model

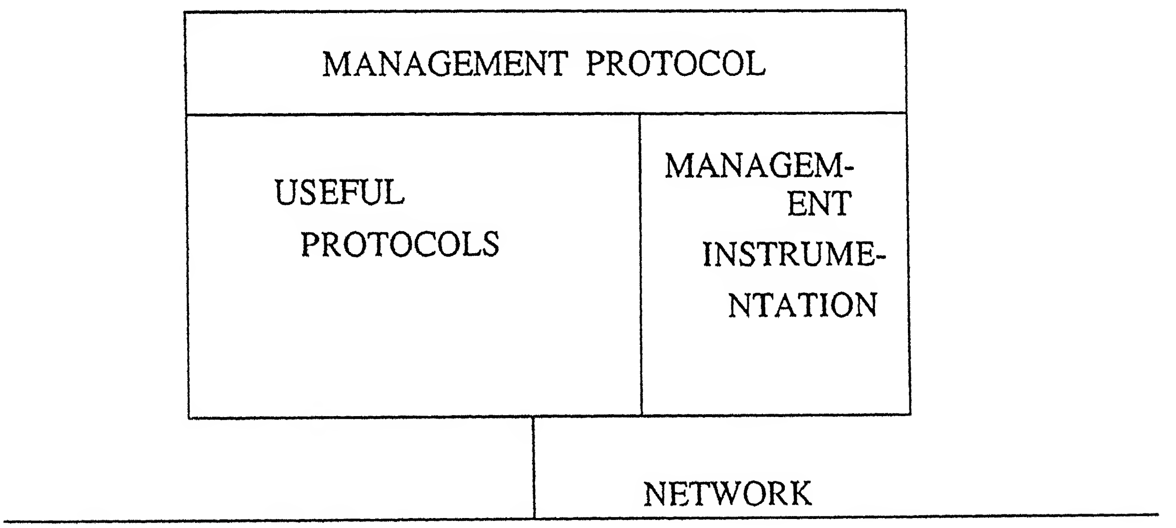

Figure 3.2: Components of a managed node

3.2.2 Managed Node

A managed node (MN) refers to a host system such as a workstation, gateway system or media device such as a bridge.

A managed node may or may not have an agent. An agent is a network management software which collects data from the node and provides them to the network management stations when asked by them. The concept of an agent is discussed in sec. 3.4 .

The commonality of all managed nodes is that they all have some network capability. A managed node having an agent can be visualized as containing following three essential components

1. Useful protocols which perform user defined functions.

2. Management protocol which permits monitoring and control of the managed nodes.

3. Management instrumentation which interacts with the implementation of the managed node in order to achieve monitoring and control.

A managed node without an agent supports only the first option. It may have a management protocol which is different from the one being used by NMS. Therefore, to manage it, the need of a proxy agent arises. The proxy agent receives the messages from NMS and translates them to messages which the managed node can understand and respond to. The concept of proxy management was described in sec. 2.3 and that of the proxy agent is described in sec. 3.6.

3.3 Management Protocol

A network management protocol is one which is used by the NMS and the nodes to exchange management information. There are several forms which this protocol might take. For example it may be used to exchange "programs" or "commands" in some paradigm.

In the Internet Standard Management Framework, a "fetch-store" paradigm [comer94] is used. This means that the node is monitored and controlled by several variables which an NMS can read or change. Thus, the management protocol needs to be supported by the definition and declaration of various variables which it will use. Also, a standard needs to be established for their declaration and definition as well as to limit their number.

The current management framework for TCP/IP networks based on SNMPv1, which was given a stable foundation on May 1991, consists of four RFCs dealing with the above mentioned issues. These four RFCs are briefly discussed below.

RFC 1155 : Structure and Identification of Management Information for TCP/IP based networks [McClo90].

> This describes the common structures and identification scheme for the definition of management information used in managing TCP/IP based networks. The descriptions of an object information model for network management along with the set of generic types used to describe management information is included. An object is defined as a variable having a name, syntax, and valid set of operations that can be performed on them. The formal descriptions of the structure are given using Abstract Syntax Notation One (ASN.1) [barari93]. This RFC is mainly concerned with the organisational concerns of the variables and the administrative policy followed regarding assigning of names to objects. Definitions given in this RFC are provided in Appendix C.

RFC 1213 : Management Information Base for Network management for TCP/IP based networks. [McClo91]

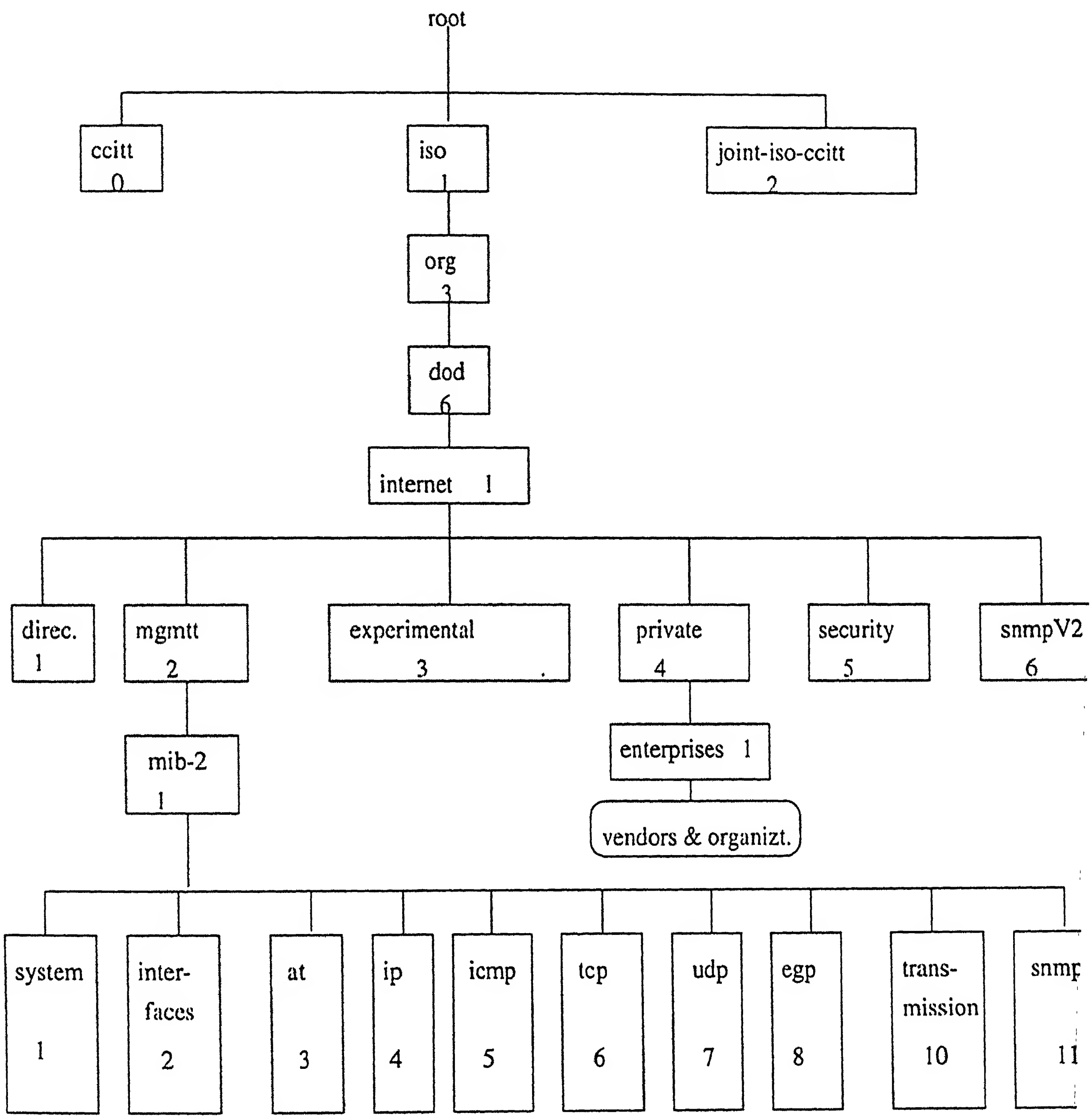

Figure 3.3: MIB tree

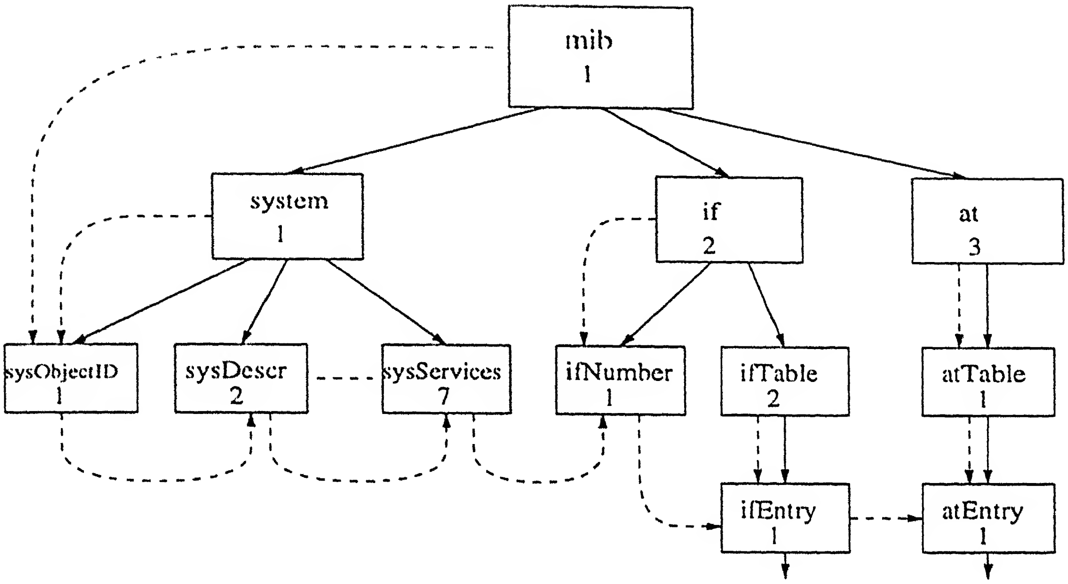

Figure 3.4: Lexical ordering of MIB objects

This RFC lists over 100 objects that hold the configuration, status and statistical information that are most needed in order to manage a system that operates in TCP/IP. These variables are kept independent of the network management protocol so as to allow vendors to incorporate software in their products that gather statistics without worrying about the protocol to be used. The objects are arranged in the following groups

System	7 objects
Interface	23 objects
Address translation table	3 objects
IP	38 objects
ICMP	26 objects
TCP	19 objects
UDP	7 objects
EGP	18 objects
Transmission	0 objects
SNMP	30 objects

ASN.1 defines a lexicographical ordering on the objects in the MIB. This ordering allows the managers to ask the agent for the set of currently available variables and to search the table without knowing the size. Two names are lexically equal if they have identical object identifiers. An example of lexical threading is shown in fig. 3.4.

RFC 1157 : A Simple Network Management Protocol(SNMP) [case90]

This RFC defines the messages that can be exchanged between a management station and a managed node to retrieve (get or get-next) or alter (set) variables values. Their operation is shown in the fig. 3.7. Thus, the number of essential management functions are limited and the introduction

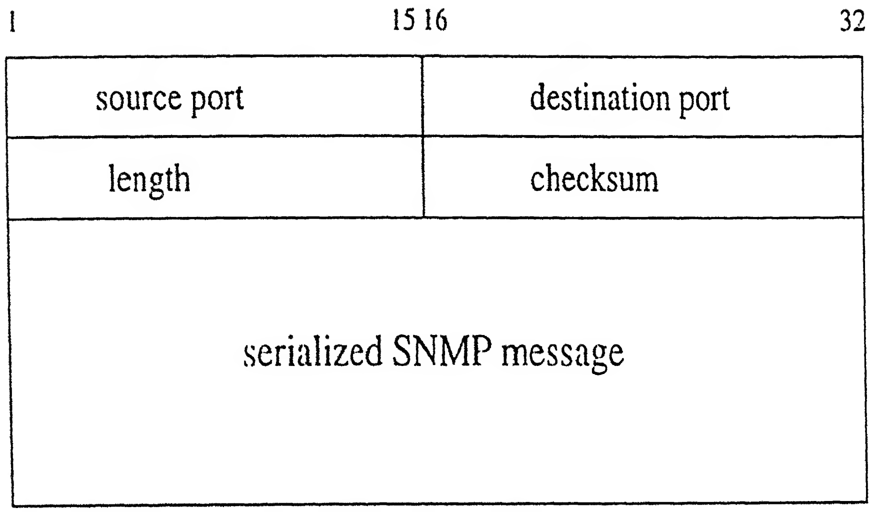

Figure 3.5: UDP packet format

of imperative commands into the protocol is avoided. The trap messages, sent by a system whose status is changing in a serious way are defined in this RFC. Moreover, it deals with the details of message format and communication protocol specification. The PDU format of various messages are given in Appendix D. This RFC deals with the communication protocol specification which is discussed in the next section.

RFC 1212 : Concise MIB definitions [mib91]
This RFC improves on the definition techniques given in RFC 1155.

3.3.1 Transport Services

SNMP is meant to be transport protocol independent. There are several mappings defined. All the mappings have one thing in common. Instances of the SNMP message data type are transmitted through a process termed as serialization. This allows an arbitrary data structure to be encoded as a sequence of octets for sending. When the octets are received, they are converted back to a data structure with identical semantics. Thus, there is unambiguous one to one mapping between the ASN.1 data structure defined above and string of bytes. All implementations of SNMP are required to accept messages which are serialized in 484 octets or less. There is no requirement on sending SNMP entities.

Mapping onto UDP is preferred. A sending SNMP entity serializes an SNMP message and sends it as a single UDP datagram to the transport address of the receiving SNMP entity. The UDP packet format is shown in the fig. 3.5.

The transport address consists of an IP address and a UDP port. All SNMP agents listen on UDP port 161. If a message contains a trap, the receiving SNMP process listens on port 162. By convention, all responses are sent with the source/destination ports swapped from the corresponding request. Thus, a get-request sent from port 1001 will have the corresponding get-response sent back at port 1001 of the same machine

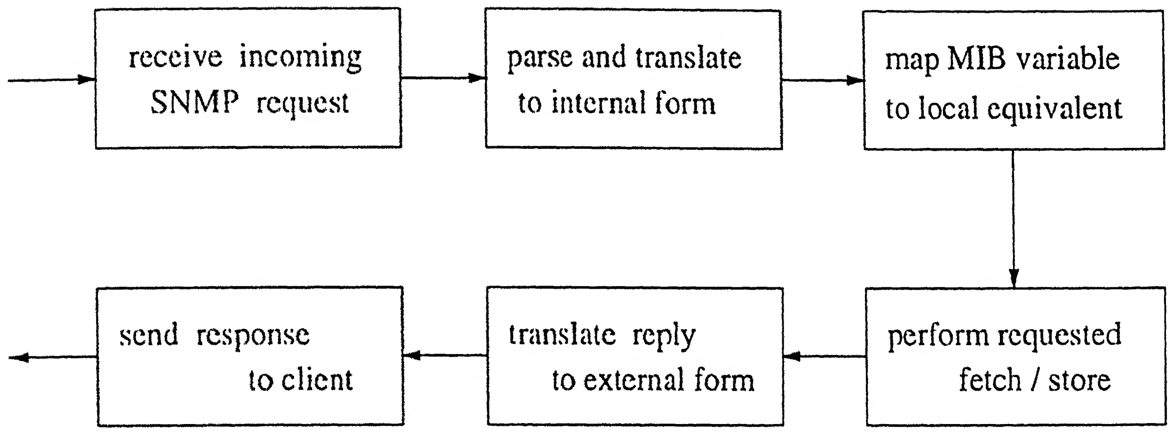

Figure 3.6: The flow of an SNMP request through a server

3.4 SNMP Agent and Manager

An SNMP agent (also called server) [comer94] is a software on the managed node which implements the internet management framework. It accepts an incoming request, performs the specified operation and returns a response. The agent repeatedly waits for an incoming request. On receiving a request, the agent first parses the message and translates it to an internal form. It then maps the MIB variable specification to the local data item that stores the needed information and performs the fetch or store operation. After that it translates the reply from internal form to external form, and sends back to the address from which the request came.

An SNMP manager is a software running on management stations. The manager forms a request message, sends it to the agent and then waits for the reply. The manager also translates the message from internal form to external form before sending, and external form to internal form after receiving. On receiving the reply, it verifies if the response matches the request. It also implements timeout and if needed, retransmission.

3.5 Protocol interactions

SNMP is an asynchronous request/response protocol which means an SNMP entity need not wait after sending message. It can continue with sending other messages or do other activities. This is due to the fact that an unreliable datagram transport protocol is used for SNMP. Due to this reason, messages can be lost and it is the responsibility of the application to do retransmission. The four basic protocol interactions are shown in fig 3.7. Of these, get and get-next is implemented for agents and set for proxy agent in this implementation.

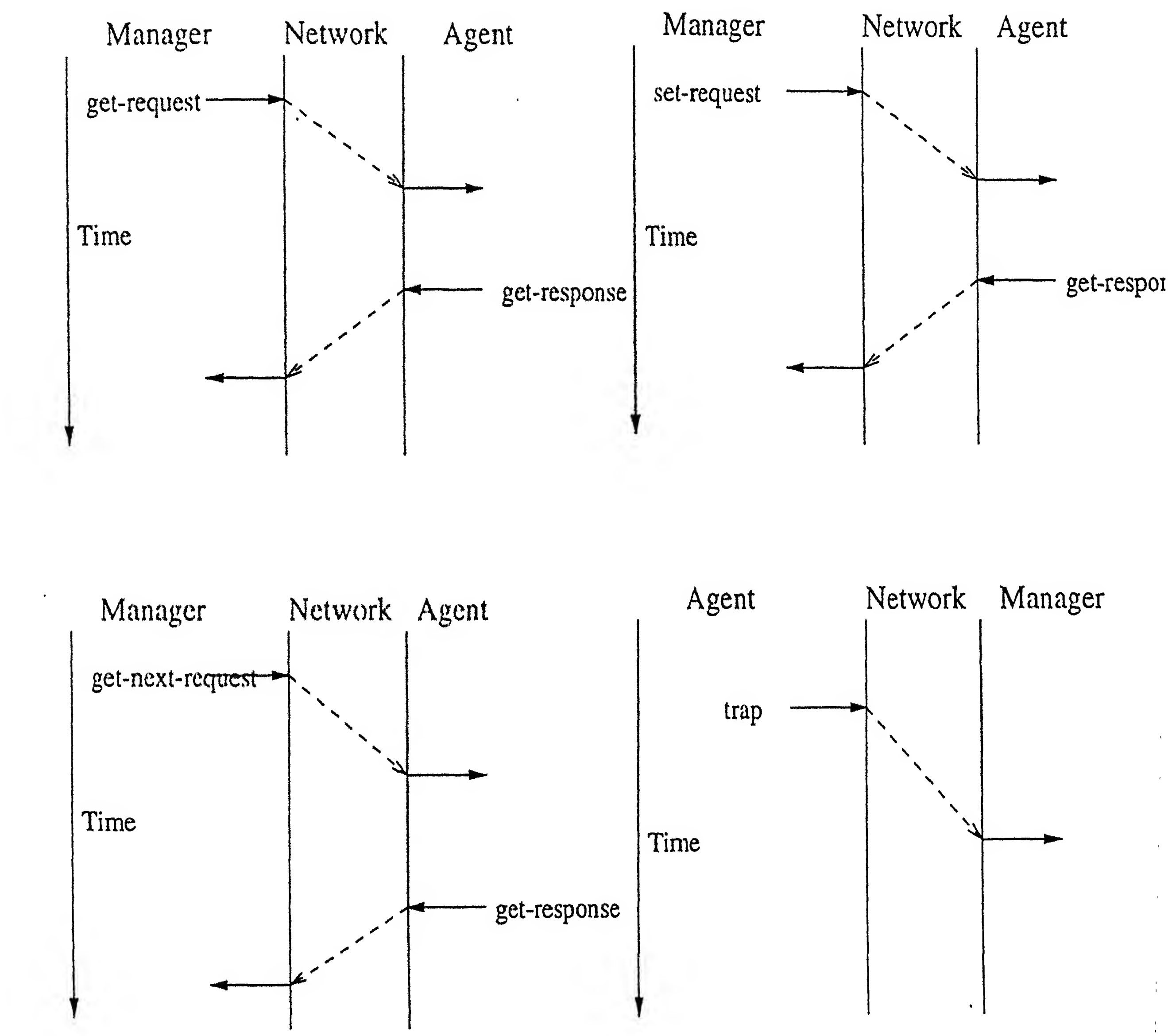

Figure 3.7: Four basic protocol interactions

0 7	8 15	16 31
type	code	checksum

Figure 3.8: ICMP packet format

3.6 Proxy Agent

Some of the hosts to be managed may not have an SNMP agent or they may be media-specific hosts. A network manager manages these hosts via a proxy-agent. A proxy agent may translate the protocol or just send messages to get management data. In this implementation, proxy agent uses a ping subroutine to get the status of the host irrespective of the fact whether an SNMP agent is running on it or not. The ping subroutine uses ICMP echo request packets. A brief description of the same follows.

Internet Control Message Protocol (ICMP) is a protocol at the IP level, giving a low level feedback about the general health of the IP level. ICMP uses delivery services of IP. If the protocol field of an IP datagram is 1, the user data contained in the datagram is an ICMP packet. The first 32 bits contains the same three fields for all messages. The three fields are

1. Type : Identifies which control message is being sent.
2. Code : Identifies a basic parameter for the control message.
3. Header Checksum : A one's complement arithmetic sum computed over the entire ICMP packet.

Some of the control messages supported by ICMP are

1. Destination Unreachable : Local IP received a datagram which it cannot route.
2. Time Exceeded : Time to live field in IP header expired.
3. Parameter Problem : Erroneous IP header.
4. Source Quench : To report a network device discarding datagrams due to lack of resources.
5. Redirect : Originated by the gateway to report another gateway closer to the destination IP address.
6. Echo/Echo Reply : To test reachability of an IP address, an echo message is sent and the receiving entity sends an echo reply message in return.
7. Timestamp/Timestamp Reply : To sample the delay in the network between two network devices.

3.7 Summary

The management framework for SNMP consists of four RFCs. These are RFC 1155, RFC 1157, RFC 1212 and RFC 1213. These are discussed briefly in this chapter. The present implementation fully conforms to the said framework. The management model is also briefly described. The communication between SNMP agents and manager is done via connectionless transport protocol such as UDP, so as not to burden the network with management traffic. The SNMP agent on receiving a request decodes it and puts in the required values and sends it back to the manager after decoding. The functions done by the agent in this procedure are also discussed.

Though not part of the framework, proxy management is discussed; this is because the implementation uses this and it needs to be clearly understood before proceeding towards implementation. The next chapter describes the implementation in detail.

Chapter 4

IMPLEMENTATION OF THE NETWORK MANAGER

4.1 Introduction

In this chapter the implementation of an SNMP manager and a proxy agent are discussed. This includes the architecture of the manager and various algorithms which are designed for implementation. The manager is implemented on MS-Windows environment. Thus, various capabilities of MS-Windows like easy to design graphical user interfaces and network support are fully utilized. The proxy agent is implemented in Linux environment. Linux is used for the first time in the ERNET Lab as a development environment. A brief description of this, is provided in the Appendix B. As a part of the thesis work, Bansal's SNMP manager and proxy agent on MicroVax was ported to Linux. This is discussed briefly in the next section.

4.1.1 Porting SNMP manager to Linux

Bansal [bansal94] implemented an SNMP manager on MicroVax with a good user interface. He also developed a proxy agent for polling hosts without an SNMP agent. As a part of this thesis work, Bansal's SNMP manager with all its user interface capabilities is successfully ported to Linux. The reason behind porting the software to Linux is that Linux is fast developing into a popular operating system for PCs. It has all the capabilities of UNIX operating system[1]. Also since its source code is readily available and is for free, it is fast becoming a popular development environment as developers can change the operating system to suit their needs.

In porting the software, the algorithms developed by Bansal were followed and programs were either rewritten or modified. Here, the communication between the manager and the proxy agent is carried out using socket approach. The major changes in the implementation are changes in the functions for sending and receiving SNMP messages from proxy agent and dynamic allocation of memory. These are done as the working of "signal" function in Linux is only partly supported and the amount

[1] http://sunsite.unc.edu/pub/linux

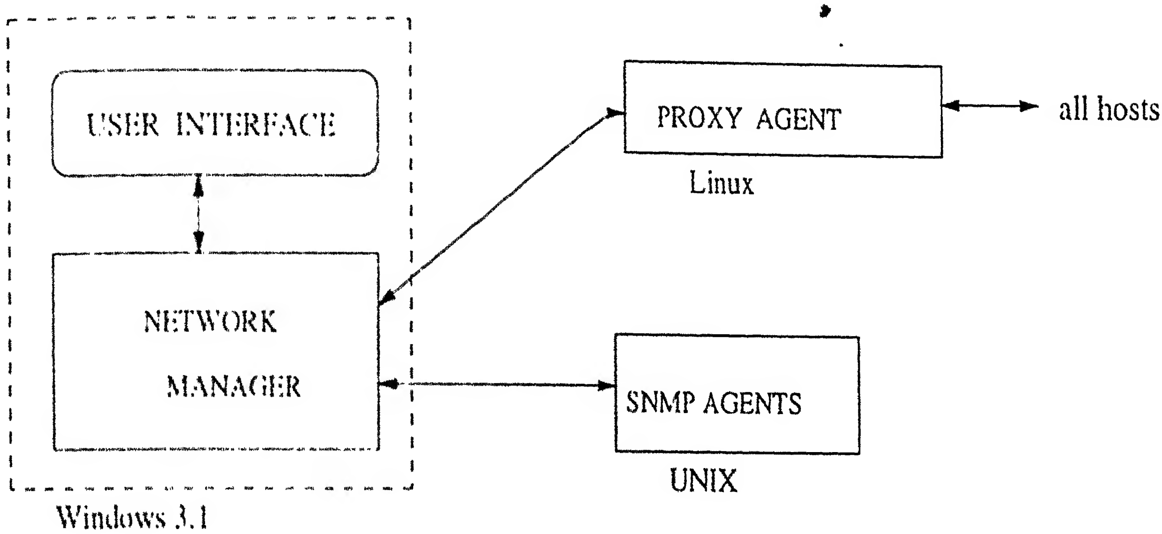

Figure 4.1: Block diagram of implementation

of memory allocated to a process is limited. As far as the proxy agent is concerned, in addition to sorting out the above problems, the ping routine is modified to run on Linux. The changes are needed because ICMP options are different and some of them are not supported in Linux and structures and variable names as defined in header files are different.

The next section discusses architecture of the implementation of SNMP manager on Windows environment.

4.2 Architecture

4.2.1 Introduction

This section explains the architecture of the management framework used in the implementation of the network manager with user interface on MS-Windows. This includes algorithms used and the architecture of the network manager and the proxy agent. Whereas the network manager resides on Windows 3.1 environment, the proxy agent runs on 4.3 BSD compatible Linux System. The proxy agent is used to monitor the status of all hosts. The network manager can poll directly a host which is running an SNMP agent for any MIB value. The block diagram representation of the network manager along with proxy agent and SNMP agents is shown in fig. 4.1 . The figure also shows the environments on which these are implemented.

The network manager performs two functions. First it polls the proxy agent to get the status of hosts in the network. The proxy agent sends an ICMP echo request packet to the host and waits for reply. Once it receives the status of the host, the proxy agent forms an SNMP reply and sends back to the manager. Secondly, it polls any host which is running an SNMP agent for MIB values. This is done by sending an SNMP get or get-next request. The user interface shows the current status of the network. The algorithms for network manager and proxy agent are shown in the figure.

In this implementation interprocess communication between different environments is used. This done using winsock library functions [martin92].

4.2.2 Network Manager

The manager has two functions as stated above. In both the cases, ie. polling the proxy agent and querying SNMP agents for MIB values, it converts a C structure into ASN.1 data types. Using Basic Encoding Rules (BER), it encodes the ASN.1 data type for sending it to the proxy agent or the SNMP agent. For sending the message to the proxy agent and different hosts from from the manager, the program uses "winsock" [martin92]. Also, the graphical capabilities of windows are made use of, like drawing lines, writing text with different background colours. The message driven program control of windows is utilized fully in the program. The windows own memory allocation routines are used in the programming. Thus, windows allocate memory to the process and memory management of windows is made use of [conger94].

The manager does the following things before displaying the interface(fig. 4.2).

1. It calls "WSAStartup()" which allows the application to specify the version number and to retrieve the details of the implementation.

2. It reads the configuration file "config.txt" and fills up the agent structure for all the hosts. The configuration file contains three fields namely, host name, IP address and whether its a SNMP agent or not. The last field is an integer where "one" denotes an SNMP agent and "zero" specifies its absence. The number of hosts is obtained by calculating the number of structures successfully filled.

3. It loads the IP address of the machine on which it is running. This is different from UNIX implementation where with the help of "getlocalhost()" one can get the local host name. This function is not supported on the Winsock used. The IP address of the proxy agent is loaded into a string.

4. The socket for communication with the proxy agent is opened and bound with IP address. The port number used is 1000.

5. To provide non-blocking operation of sockets, "WSAAsyncSelect()" is used which specifies that in case any incoming data on that socket, the "WM_USER2" window message routine should be processed.

"WSAAsyncSelect()" helps in the non-blocking operation of the socket. The program can continue with other processing while waiting for packets to be received at the receiver port. This saves a lot of processing time which was previously getting lost in waiting.

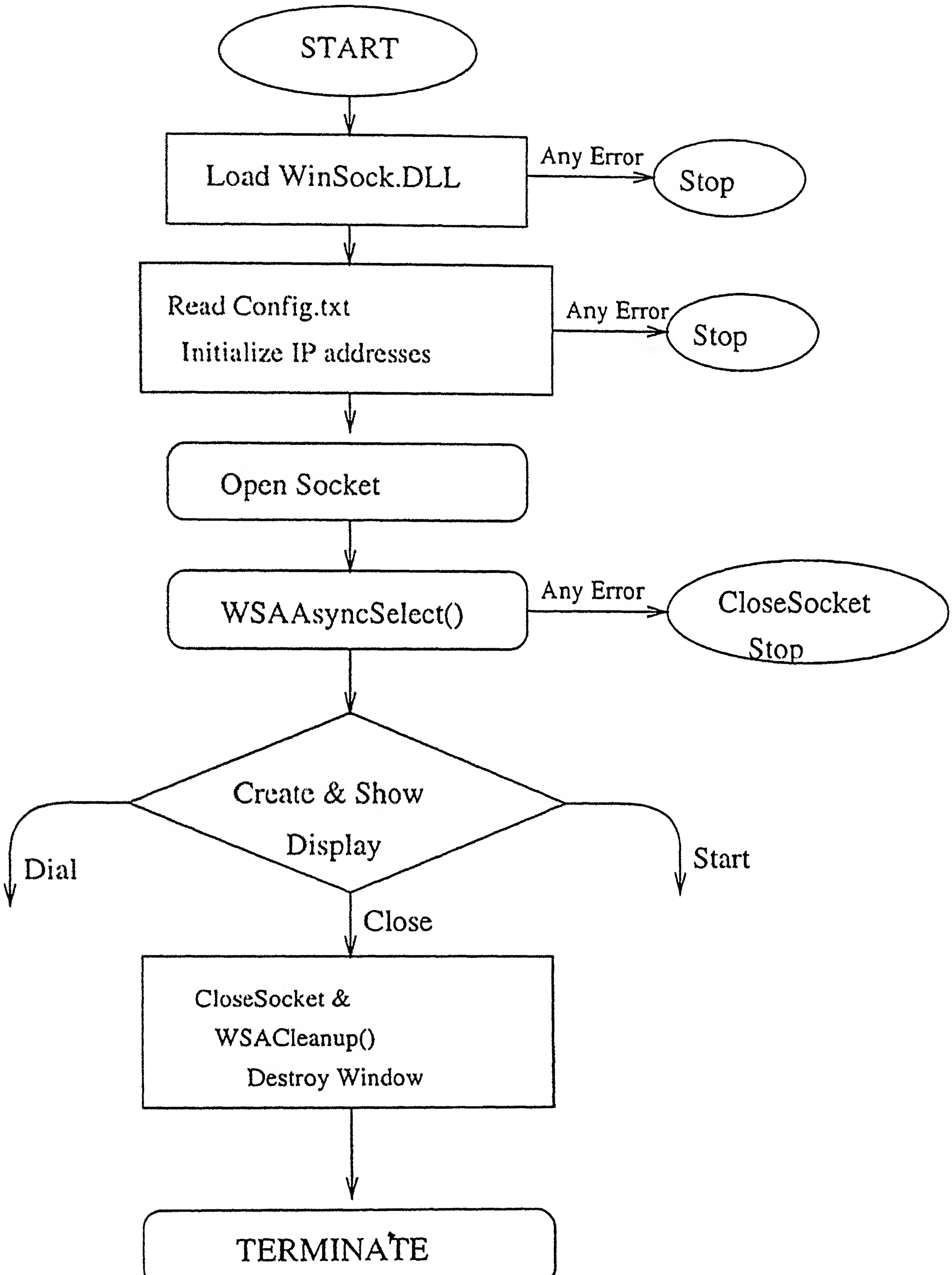

Figure 4.2: Flow chart of the manager

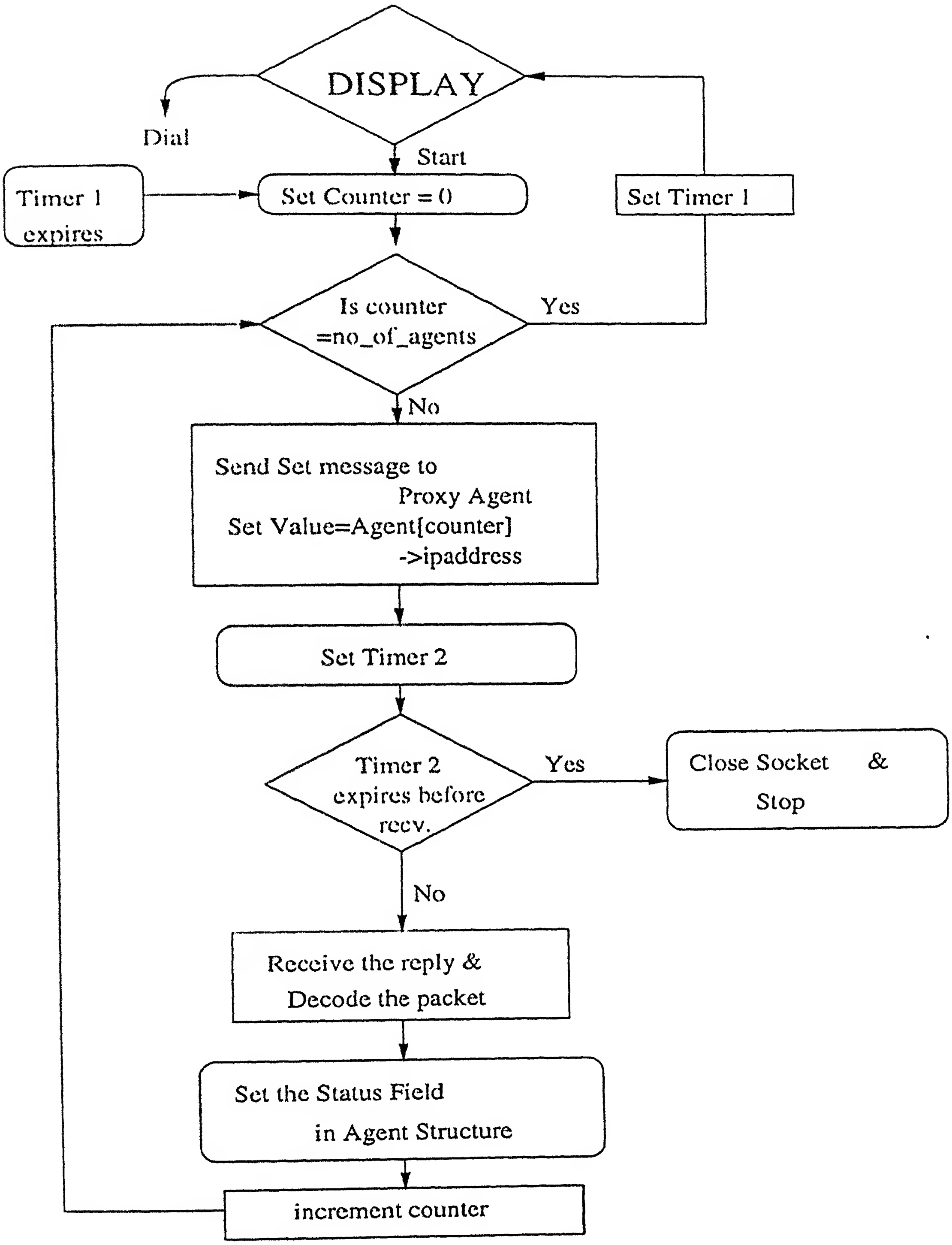

Figure 4.3: Flow chart of manager(contd.)

For polling the proxy agent, the manager does the following(fig 4.3)

1. Resets the counter to zero and start processing "WM_USER1" window message routine.

2. The manager checks the value of counter, loads the IP address of the machine, to which ICMP echo packet is to be sent, in the "set value" puts the pdu type "SET" and forms the SNMP packet with the coder_snmp() function.

3. After sending the packet to proxy agent it sets a timer. If reply is not received from proxy agent within the timeout period, the program terminates.

4. The packet returned by the proxy agent is decoded. The value in the var bind structure specifies the status of the host to which ICMP echo packets were sent. Zero specifies status down and One specifies status up.

This procedure is continued till all the hosts are polled in this manner. After this the display is shown with the recent status of hosts. The polling of hosts via proxy agent is repeated after every 5 minutes in the present implementation.

The manager can also send the SNMP request to hosts running SNMP agent. Another capability of windows, "Dialog Box", is utilized here. It provides an easy way for inputing data to a program. The manager makes an SNMP packet, sends to the host and waits for the reply, this it does in the following steps (fig. 4.4).

1. The manager takes the host name, sub-oid and pdu type from dialog box.

2. Opens another socket with port number 1001. Then it codes the message and sends it to the agent on UDP port 161.

3. The same non-blocking operation is used and if reply comes within timeout period, it is decoded. The value is shown in a "message box".

4. The socket is closed with "closesocket()" call.

The above discussion lists the working of the manager. In this discussion the term manager apples to the process running on the network manager end. The flow charts are provided showing the program flow. In the next section user interface is explained.

4.2.3 User Interface

The user interface is developed to provide user friendly controls and easy interaction with the program. The interfaces are shown in fig. 4.5 and fig. 4.6.

The display begins with a message saying "KINDLY WAIT". At this moment one can either start polling the proxy agent or else can send a request to an SNMP agent. To start the polling, the user clicks the menu item "Start". After the polling is over, a message box appears on the screen with a message "polling over" and screen is refreshed with a display showing status of hosts in the network. Any error occurred during polling is shown via message box and if proxy agent is not responding the program terminates.

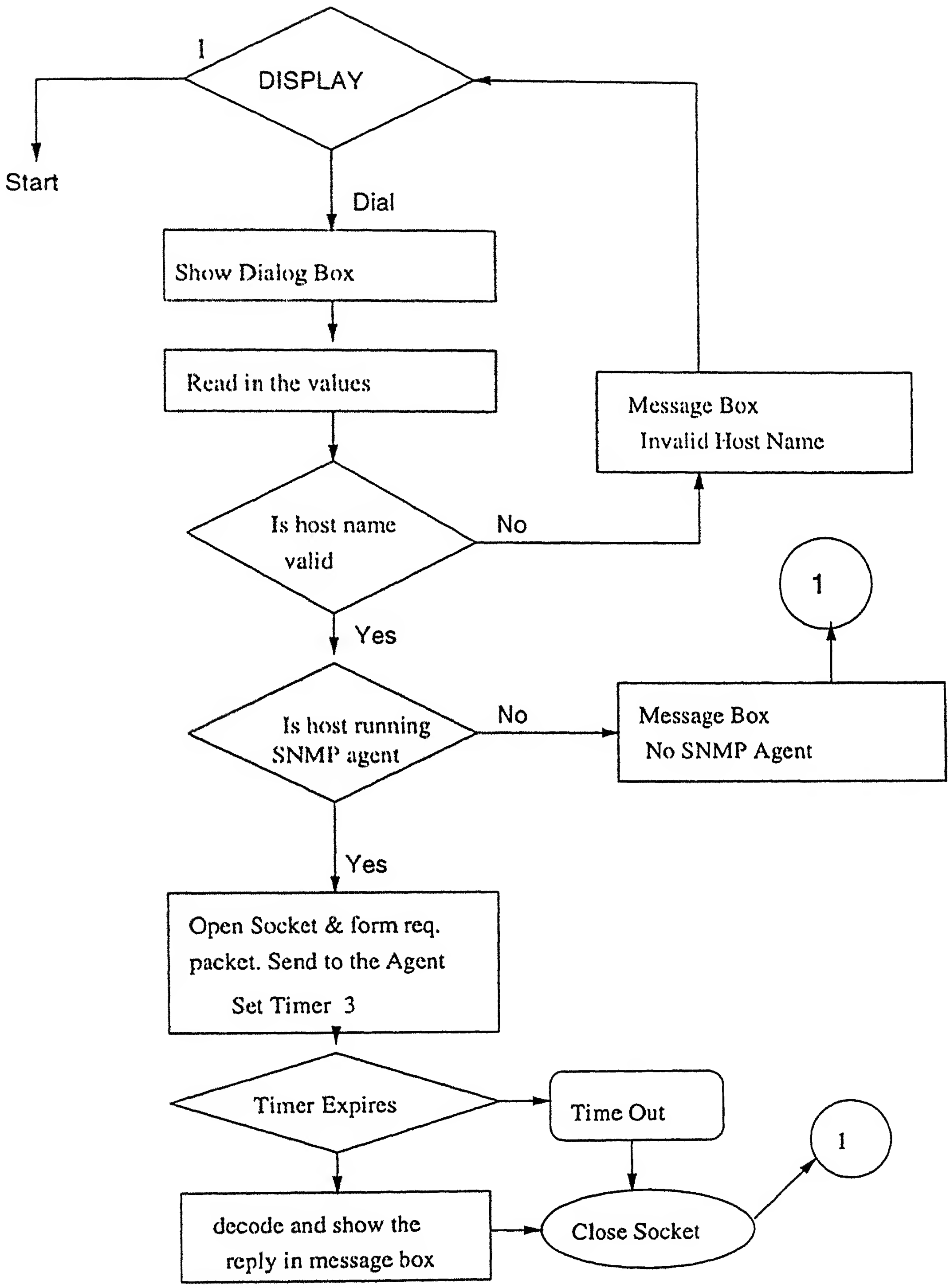

Figure 4.4: Flow chart of manager(contd.)

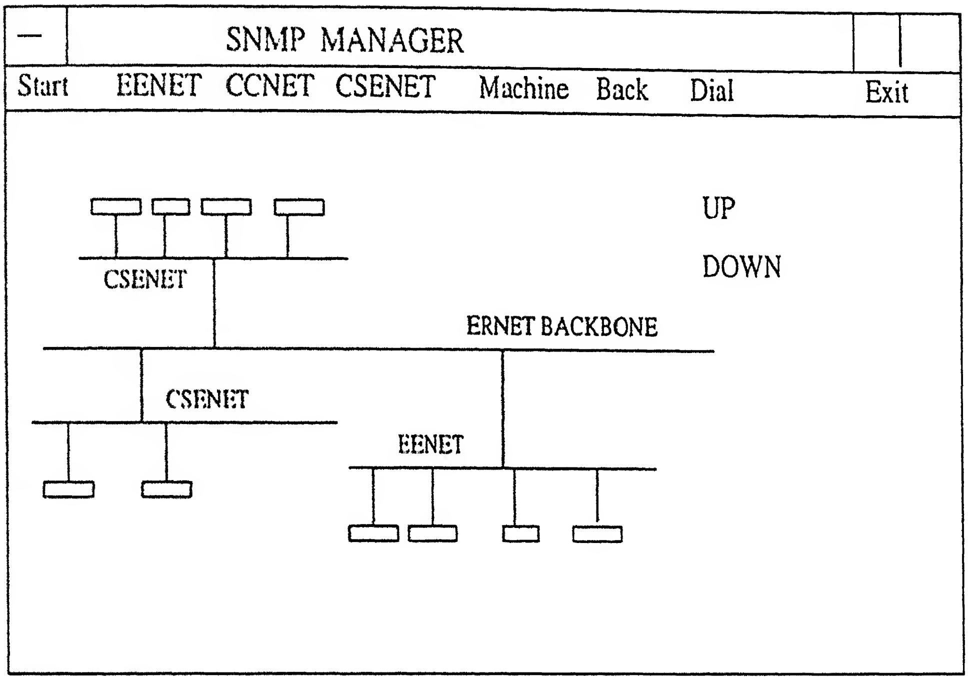

Figure 4.5: Main display

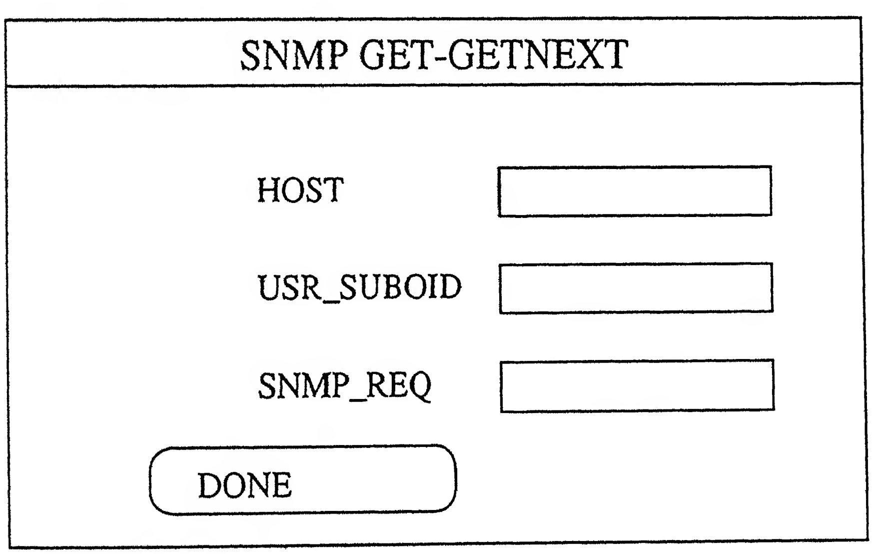

Figure 4.6: Dialog Box

This is done, so as not to hang up the program before one realizes that the fault lies with the proxy agent.

The individual displays for EENET, CCNET and CSENET can be got by clicking respective menu items. In this display, the respective IP addresses of the hosts displayed are shown along with. The menu item "Back" brings the display back to the original one.

To send an SNMP request to a host, one has to click menu item "Dial". This opens a dialog box. The dialog box provides a user friendly way of sending an SNMP request. The sample values are shown along with it for the convenience of the user.

The cursor position can be moved either by pressing "Tab" key or else clicking left mouse button at each of the item. After giving the value of sub-oid, specifying host and the request pdu type, the "DONE" button is clicked. At this an SNMP message is sent and the reply from the host is shown in a message box. The "SET" operation is not supported by the manager.

The polling of the hosts via proxy agent is done after a regular interval hence the screen is refreshed from time to time. The "Dial" is disabled during the period when proxy agent is being polled. When user clicks "Exit" menu item, the implementation frees all the memory and closes the socket which was opened for communication with proxy agent. Then manager issues a "WSACleanup()" to terminate the use of windows sockets DLL.

In this implementation, the dynamic allocation of memory is done and the memory is deallocated whenever possible. This is done to keep the memory requirement of the program to the minimum.

4.2.4 Proxy Agent

The status of the hosts is found out via proxy agent which uses ICMP echo messages. This was developed by Bansal [bansal94] by modifying "ping" routine. It was further modified to run on Linux. The algorithm followed is shown in fig. 4.7 .

The program opens port 1000 and waits for requests from the manager. Once the proxy agent receives an SNMP request, it decodes the packet and finds out the host IP address from the value field of the sub-oid "1.1.0". The proxy agent then sends ICMP echo requests packets to this host. A modified version of "ping" routine is used for this purpose. If the reply is received within the timeout period, the status of the particular host is "UP" else it is "DOWN". After this reply is sent with recent status in the value field of the sub-oid. The proxy agent then goes into the waiting mode.

4.3 User Defined Structures and Functions

In this section, the various structures and functions used in the program are described below. The coding/encoding functions used by the manager and the proxy agent are developed by Barari [barari93].

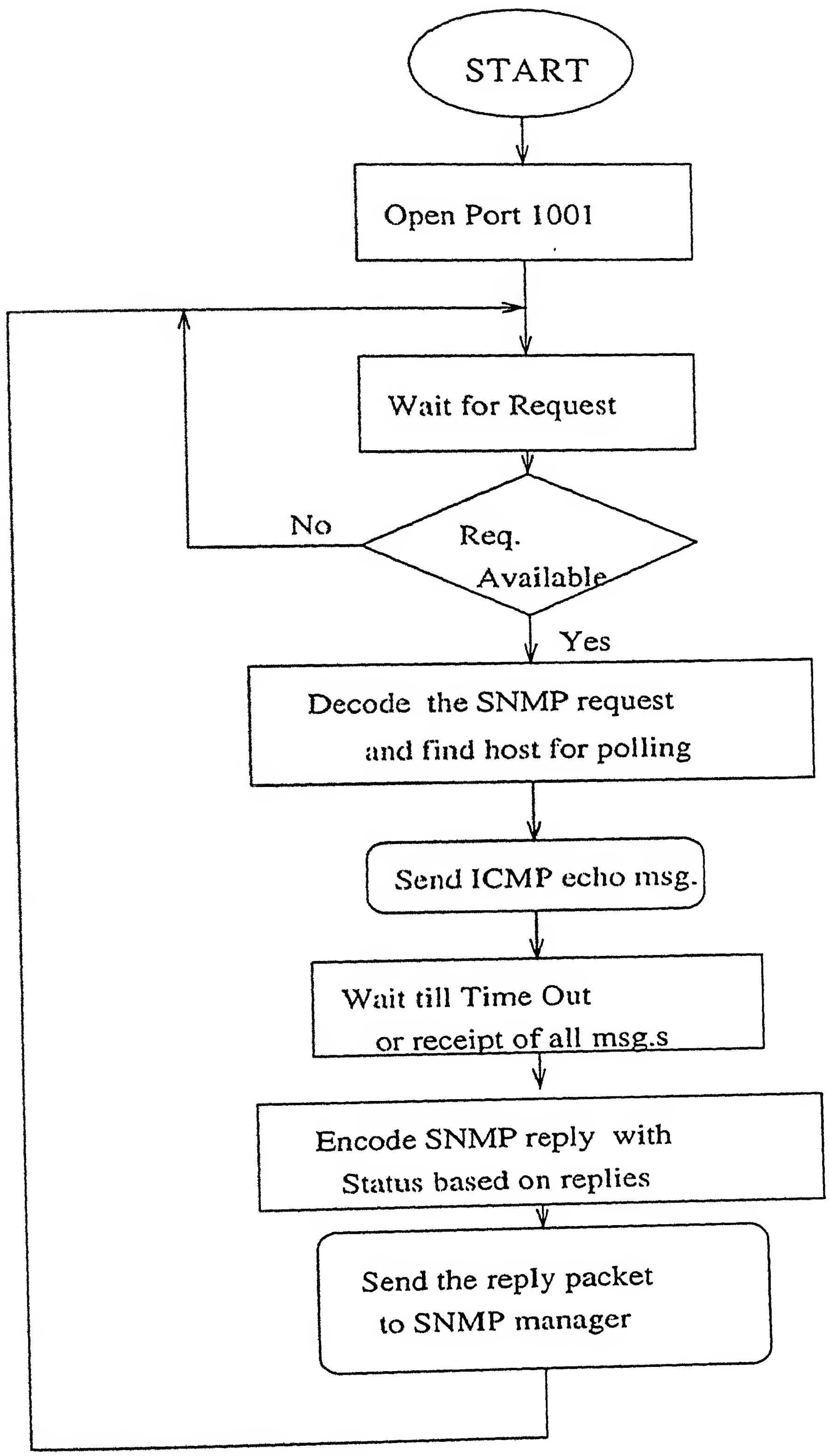

Figure 4.7: Flow chart of Proxy Agent

4.3.1 ASN.1 Encoding/Decoding Structures

A request descriptor structure is used to store all information about one SNMP message.

```
struct req_desc {
        u_char reqtype;             /* request type */
        u_char reqid[10];           /* request descriptor */
        int reqidlen;               /* length of identifier */
        u_char err_stat;            /* error status */
        u_char err_idx;             /* error index */
        int err_stat_pos;           /* place of err stat */
                                    /* in the message */
        int err_idx_pos;            /* place of err_idx */
        int pdutype_pos;            /* position of pdu type */
        struct snbentry *bindlf; /* beginning of binding list */
        struct snbentry *bindle; /* end of binding list */
            };
```

Each snbentry structure stores information about one variable binding in the message. The list is doubly linked to form a queue of the variable bindings.

```
struct snbentry {
        struct oid sb_oid;          /* object id in the internal form*/
        struct snval sb_val;        /* value of the object */
        int sb_a1slen;              /* length of the ASN.1 string */
        struct snbentry *sb_next; /* next node in the list */
        struct snbentry *sb_prev; /* previous node in the list */
            };
```

The structure "oid" stores the sub object identifiers.

```
struct oid {
        int id[32]; /* array of sub-identifiers */
        int len;    /* length of this object id */
            };
```

The structure "snval" stores the value of the object identifier.

```
struct snval {
        u_char sv_type; /* variable type */
        union {
            long sv_int; /* variable is one of integer */
                         /* counter, gauge, timeticks */
            struct {
                char *sv_str; /* string's contents */
                int sv_len;   /* string's length */
                  } sv_str;
            struct oid sv_oid; /* variable is an OID */
            IPaddr sv_ipaddr; /* variable is an IP address */
              } sv_val;
```

```
        };
IPaddr is typedefined as
typedef u_char IPaddr[4];
```

4.3.2 Agent Structure

For each agent there is a structure "agent" which stores information (based on entries in the "config.txt" file) pertaining to it.

```
struct agent {
        char *name;   /* domain name of the host */
        char *ipaddr; /* ip address in a.b.c.d form */
        int status;   /* status of the host */
        int a_snmp;   /* whether it runs SNMP agent */
            };
```

There is a global array of pointers to the structure "agent", each array entry for each agent.

4.3.3 Functions in the Program

The various functions used by the manager and the proxy agent are discussed below.

File Read Functions

config()
This function reads each host entry from "config.txt" file. It allocates memory to the agent structure and fills it with entries for a host in the file. It terminates the program if any error occurs.

Display Function

DrawLine()
This function draws a line between two points whose X and Y coordinates are given.

WriteText()
This function writes the specified text at the specified position with the background black or white as given by the variable "col". If it is "0", text is white on black background and it is "1", it is otherwise.

Set()
This function gives X and Y coordinates of host display depending upon its IP addresses' tenth character which is different for different sub-nets.

SetRout()
This function gives X and Y coordinates of router display depending upon its IP addresses' fourteenth character.

check()
This function checks whether a SNMP agent is running on the host or not and displays a message to that effect.

Memory Allocation Functions

acalloc()
This function allocates specified number of bytes from windows local heap and locks it.

afree()
This function frees the memory allocated by "acalloc()".

Encoding/Decoding Functions

a1readlen()
This function reads and returns the length of an ASN.1 encoded object type. It also returns the length of the length field of an ASN.1 encoded object through a pointer passed to it.

a1writelen()
This function writes the length of an object in ASN.1 encoded form and returns the number of bytes in the encoded length field.

a1readint()
This function converts an ASN.1 encoded integer into a internal form. It requires the length, in number of bytes, of the value field of the encoded integer as an input and the byte stream consisting the encoded integer.

a1writeint()
This function writes the value of the integer in ASN.1 encoded form and returns the length of the ASN.1 encoded integer in number of bytes.

a1readoid()
This function converts an ASN.1 encoded object identifier into internal form. It removes the fixed prefix "1.3.6.1.2.1" common to all SNMP object ids and then stores the rest in the structure "oid". The length of the object id, the encoded byte stream and pointer to the "oid" structure must be passed to it. It fills the "oid" structure and returns 1 in case of no error; else it returns -1.

a1writeoid()
This function converts object identifier into ASN.1 encoded form and returns the length of the encoded object identifier in number of bytes.

a1readval()
This function reads the value of the instance "value" from ASN.1 encoded byte stream and stores it into an "snval" structure. It returns 1 in case no error and -1 in case of error.

a1writeval()
This function encodes the value of "value" of an ASN.1 variable binding from internal structure to byte stream and returns the length of the value in number of bytes.

sna2b()
This function converts ASN.1 encoded binding into internal form and filling the elements of the "snbentry" structure. In case of any error -1 is returned.

snb2a()
This function converts the list of variable bindings in internal form in the "snbentry" structure to ASN.1 form and store byte stream in "sb_a1str" element in the same structure. In case of any error -1 is returned else 1 is returned.

snparse()
This function converts the ASN.1 encoded into internal form, filling the "req_desc" structure. In case of any error -1 is returned else 1 is returned.

mksnmp()
This function makes an ASN.1 encoded SNMP packet from a "req_desc" structure. It returns the length of the packet in number of bytes.

snmp_free()
This function is used to free any memory allocated to a "req_desc" structure.

snfreebl()
This function frees any memory allocated to the doubly linked list of "snbentry's".

coder_snmp()
This function gets the type of packet and list of sub-object identifiers. Then fills the "req_desc" structure and calls above encoding routines to form an ASN.1 encoded SNMP message. It returns the length of the packet in number of bytes.

decoder()
This function decodes an ASN.1 encoded SNMP message and fill up a "req_desc" structure. It returns 0 if there was no error in decoding else returns -1.

Functions to Show Error Messages

wsa_error()
This function creates and shows a message box indicating the particular error which has

occurred in case of winsock errors. The error occurred is known by calling WSAGetLastError() function.

mem_error()
This function creates and displays a message box indicating memory fault in case windows could not allocate memory to it.

Function to Send Ping Packet

send_ping_packet()
This function is based on user program "ping". The arguments passed to this function are host address and number of packets to be sent. ICMP echo packets are sent based on these arguments. The function returns the number of reply packets received in the response.

4.4 Summary

In this chapter implementation details including functions used and user defined structures are described. This chapter aims at clearly specifying the implementation. In addition to this, programs are documented in a way so as the reader can clearly get what the program is doing. "README" and "doc" files are provided as online help to users and developers both. The next section concludes the work and discusses future scope of the work.

Chapter 5

CONCLUSION

The primary objective of this thesis was to implement an MS-Windows based network manager with user interface. This was achieved with the implementation of the network manager, which shows the status of local network and also can get MIB values from SNMP agents, on Windows 3.1 using winsock. The implementation has two parts.

First the network manager program was written by building upon the work on implementation of BER using ASN.1 [barari93]. The present implementation has following features.

Polling the status of hosts using a remote proxy agent.

Querying MIB values from hosts with SNMP agents.

Displaying status of the local network.

Menu driven user interface.

The present implementation is having proxy agent running on Linux.

Secondly, the proxy agent developed by Bansal [bansal94] was modified to run on Linux and communicate with the network manager running on MS-Windows. Thus, this an example of interprocess communication between processes running on different environments. The implementation done can be applied to larger networks as well.

The source files for the network manager are stored in "c:"bhar"fine" in the PC as well as submitted to the "ERNET Lab" in floppy. The executable file is also stored along with and can be run from an icon in the program group "Applications". The source files and executable file (proxy) of the proxy agent are stored in the directory "/home/bhar/arch/prox".

As a part of this thesis work, the network manager residing on MicroVax was successfully ported to Linux. The source files of the ported program are in the directory "/home/bhar/arch/mgmt".

5.1 Scope for Further Work

Larger networks can be managed using present implementation with minor modifications. The proxy agent concept can be modified to get a local manager for local network, which in turn can be polled by a central network manager, thereby reducing the management traffic in the network. This concept can be very helpful in managing

a large network as then the manager will be polling a few local proxy managers rather than all the hosts.

The manager can be ported to X-windows which has better multiprocessing capabilities than MS-Windows. This can be a future scope of work.

By modifying the program, MIB values can be polled continually and a local database at the manager end can be maintained for ready reference and analysis.

Trap is not recognized by the present implementation. Trap is issued by the agent when an exception occurs. The manager can then poll the agent to know about the exact status.

The present implementation, though exploiting many features of MS-Windows, does not provide multiple windows so that each subnet can be viewed simultaneously. This can be added to the present implementation.

With the enhancements as suggested, the present implementation can be used for complete network management of any network irrespective of the size.

Bibliography

[comer91] Douglas E. Comer. *Internetworking with TCP/IP Vol I.* Prentice Hall of India. 1991.

[comer94] Douglas E. Comer, David L. Stevens. *Internetworking with TCP/IP Vol II.* Prentice Hall of India. 1994

[rose91] Marshall T. Rose. *The Simple Book -An Introduction to Management of TCP/IP based Internets.* Prentice Hall. 1991

[case90] J. Case, M. Fedor, M. Schoffstall, J. Davin. *RFC1157- A Simple Network Management Protocol* IETF . 1990

[McClo90] M. Rose, K. McCloghrie. *RFC1155-Structure and Identification of Management Information for TCP/IP based Internets* IETF . 1990

[McClo91] K. McCloghrie, M. Rose. *RFC1213-Management Information Base for Network Management of TCP/IP based Internets* IETF . 1991

[mib91] M. Rose, K. McCloghrie. *RFC1212-Concise MIB Definitions.* IETF . 1991

[fiet95] Sidnie Fiet. *SNMP A Guide to Network Management.* McGraw Hill . 1995

[martin92] Martin Hall, Mark Towfiq, Geoff Arnold, David Treadwell, Henry Sanders . *Windows Sockets-An Open Interface for Network Programming under Microsoft Windows* 1992

[bapat92] Bapat V. K. *Studies in implementation of SNMP.* M. Tech Thesis IIT. Kanpur 1992

[barari93] Barari T. *Implementation of MS-DOS based SNMP Manager.* M. Tech Thesis IIT. Kanpur 1993

[bansal94] Anup Bansal. *A Network Manager based on SNMP for TCP/IP networks.*M. Tech Thesis IIT. Kanpur 1994

[dallas88] I. N. Dallas, E. B. Spratt. *issues in LAN management.* north-holland . 1988

[conger94] Jim Conger *Windows Programming Primer Plus* Galgotia Publication . 1994

[INM4] Adarshpal S. Sethi, Yves Raynaud, Fabienne Faure-Vincent. *Integrated Network Management IV*. Chapman & Hall .1995

Appendix A

WINSOCK

The Windows Sockets specification defines a network programming interface for Microsoft Windows, which is based on the "socket" paradigm as stated in the Berkeley Software Distribution (BSD). It encompasses both familiar Berkeley socket style routines and a set of set of Window-specific extensions designed to allow the programmer to take advantage of the message driven nature of Windows.
Network software which conforms to this Windows Sockets specification are considered "Windows Sockets Compliant". Specifically, all Windows Sockets implementations support both stream and datagram sockets. The Windows Sockets Specification has been built upon the Berkeley Sockets programming model which is the standard TCP/IP networking.
The Windows Sockets specification includes about all the Berkeley-style socket routines. These includes accept(), bind(), send(), recv(), etc.
The major issue in porting applications from a Berkeley sockets environment to a Windows environment is "blocking"; ie. invoking a function which does not return until associated operation is completed. The problem arises when the operation takes arbitrarily long time to complete. However, within a windows sockets implementation, a blocking operation which cannot be completed immediately is handled as follows. The DLL initiates the operation, and then enters a loop in which it dispatches any Windows messages and then checks for the completion of the Windows Sockets function. Thus message driven operation of windows is utilized. To facilitate this operation, some routines are provided which are windows specific, these are referred to as Windows-specific Extension Functions. These extended APIs allow message-based asynchronous access to network events. "Windows Sockets " document of latest version is available in electronic form, requests for the same is to be addressed to winsock-request@ ftp. com.

Appendix B

LINUX

Linux [1] is a completely free reimplementation of the POSIX spec, with SYSV and BSD extensions (which means it looks like Unix, but does not come from the same source code base), which is available in both source code and binary form. It was written by Linus B. Torvalds with help from various contributers on internet.
Linux runs only on 386/486/Pentium machines with ISA, EISA, PCI and VLB busses. Ports to other machines, including MIPS, PowerPC, and PowerMAC, are under way and showing various amounts of progress

Linux Features
multitasking: several programs running at once.
a)multiuser: several users on the same machine at once.
b)runs in 386 protected mode.
c)has memory protection between processes, so that one program can't bring the whole system down.
d)demand loads executables: Linux only reads from disk those parts of a program that are actually used.
e)shared copy-on-write pages among executables. This means that multiple process can use the same memory to run in. When one tries to write to that memory, that page (4KB piece of memory) is copied somewhere else. Copy-on-write has two benefits: increasing speed and decreasing memory use.
f)virtual memory using paging (not swapping whole processes) to disk: to a separate partition or a file in the filesystem, or both, with the possibility of adding more swapping areas during runtime (yes, they're still called swapping areas). A total of 16 of these 128 MB swapping areas can be used at once, for a theoretical total of 2 GB of useable swap space.
g)a unified memory pool for user programs and disk cache, so that all free memory can be used for caching, and the cache can be reduced when running large programs.
h)dynamically linked shared libraries (DLL's), and static libraries too, of course.
i)does core dumps for post-mortem analysis, allowing the use of a debugger on a program not only while it is running but also after it has crashed.

[1] http://www.math.uio.no/doc/linux/HOWTO/tekst/INFO-SHEET

j)mostly compatible with POSIX, System V, and BSD at the source level.
k)all source code is available, including the whole kernel and all drivers, the development tools and all user programs; also, all of it is freely distributable.
l)POSIX job control.
m)pseudoterminals (pty's).
n)TCP/IP networking, including ftp, telnet, NFS, etc.

Appendix C

RFC1155

```
RFC1155-SMI DEFINITIONS ::= BEGIN

   EXPORTS - EVERYTHING
            internet, directory, mgmt,
            experimental, private, enterprises,
            OBJECT-TYPE, ObjectName, ObjectSyntax, SimpleSyntax,
            ApplicationSyntax, NetworkAddress, IpAddress,
            Counter, Gauge, TimeTicks, Opaque;

          · the path to the root

               internet OBJECT IDENTIFIER ::={ iso org(3) dod(6) 1}

            directory OBJECT IDENTIFIER ::={ internet 1}
            mgmt OBJECT IDENTIFIER ::={ internet 2}
            experimental OBJECT IDENTIFIER ::={ internet 3}
            private OBJECT IDENTIFIER ::={ internet 4}
            enterprises OBJECT IDENTIFIER ::={ private 1}

            definition of object types

               OBJECT-TYPE MACRO ::=
            BEGIN
                 TYPE NOTATION ::= "SYNTAX" type
                      "ACCESS" Access
                      "STATUS" Status
                 VALUE NOTATION ::= value (VALUE ObjectName)
                      Access ::= "read-only"
                      | "read-write"
                      | "write-only"
                      | "not-accessible"
                      Status ::= "mandatory"
```

```
                | "optional"
                | "obsolete"
END

names of objects in the MIB

    ObjectName ::=
        OBJECT IDENTIFIER

    -- syntax of objects in the MIB
    ObjectSyntax ::=
        CHOICE{
            simple
                SimpleSyntax,
   note that simple SEQUENCEs are not directly
   mentioned here to keep things simple (i.e.,
   prevent mis-use). However, application-wide
   types which are IMPLICITly encoded simple
   SEQUENCEs may appear in the following CHOICE

          application-wide
              ApplicationSyntax
              }

          SimpleSyntax ::=
             CHOICE{
                number
                  INTEGER,
                string
                  OCTET STRING,
                object
                  OBJECT IDENTIFIER,
                empty
                  NULL
          }

           ApplicationSyntax ::=
            CHOICE{
               address
                  NetworkAddress,
               counter
                  Counter,
               gauge
                  Gauge,
               ticks
```

```
                TimeTicks,
             arbitrary
                Opaque
other application-wide types, as they are
defined, will be added here
            }
   application-wide types
        NetworkAddress ::=
        CHOICE{
           internet
              IpAddress
         }
        IpAddress ::=
           [APPLICATION 0] – in network-byte order
              IMPLICIT OCTET STRING (SIZE (4))
        Counter ::=
           [APPLICATION 1]
              IMPLICIT INTEGER (0..4294967295)
        Gauge ::=
           [APPLICATION 2]
              IMPLICIT INTEGER (0..4294967295)
        TimeTicks ::=
           [APPLICATION 3]
              IMPLICIT INTEGER (0..4294967295)
        Opaque ::=
           [APPLICATION 4] - arbitrary ASN.1 value,
              IMPLICIT OCTET STRING – "double-wrapped"
   END
```

Appendix D

RFC1157

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

            IMPORTS
              ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
                FROM RFC1155-SMI;

              top-level message

               Message ::=
                 SEQUENCE {
                   version - version-1 for this RFC
                     INTEGER {
                       version-1(0)
                           },
                   community -- community name
                     OCTET STRING,
                   data - e.g., PDUs if trivial
                     ANY - authentication is being used
                     }

            -- protocol data units

               PDUs ::=
                  CHOICE {
                     get-request
                        GetRequest-PDU,
                     get-next-request
                        GetNextRequest-PDU,
                     get-response
                        GetResponse-PDU,
                     set-request
                        SetRequest-PDU,
```

```
        trap
            Trap-PDU
        }

-- the individual PDUs and commonly used
-- data types will be defined later

GetRequest-PDU ::=
    [0]
  IMPLICIT SEQUENCE {
      request-id
          RequestID,
      error-status -- always 0
          ErrorStatus,
      error-index -- always 0
          ErrorIndex,
      variable-bindings
          VarBindList
          }

GetNextRequest-PDU ::=
    [1]
  IMPLICIT SEQUENCE {
      request-id
          RequestID,
      error-status -- always 0
          ErrorStatus,
      error-index -- always 0
          ErrorIndex,
      variable-bindings
          VarBindList
          }

SetRequest-PDU ::=
    [3]
  IMPLICIT SEQUENCE {
      request-id
          RequestID,
      error-status -- always 0
          ErrorStatus,
      error-index -- always 0
          ErrorIndex,
      variable-bindings
          VarBindList
          }
```

```
Trap-PDU ::=
    [4]
  IMPLICIT SEQUENCE {
      enterprise -- type of object generating
                          -- trap, see sysObjectID in [5]
          OBJECT IDENTIFIER,
      agent-addr -- address of object generating
          NetworkAddress, -- trap
      generic-trap -- generic trap type
          INTEGER {
              coldStart(0),
              warmStart(1),
              linkDown(2),
              linkUp(3),
              authenticationFailure(4),
              egpNeighborLoss(5),
              enterpriseSpecific(6)
          },
      specific-trap -- specific code, present even
          INTEGER, -- if generic-trap is not
                          -- enterpriseSpecific
      time-stamp -- time elapsed between the last
          TimeTicks, -- (re)initialization of the network
                          -- entity and the generation of the
                          trap
      variable-bindings -- "interesting" information
          VarBindList
          }

  -- variable bindings
  VarBind ::=
        SEQUENCE {
            name
                ObjectName,
            value
                ObjectSyntax
            }
VarBindList ::=
            SEQUENCE OF
                VarBind

  END
```

Appendix E

GLOSSARY

Abstract Syntax Notation 1 (ASN.1) A language used for defining datatypes. ASN.1 is used in OSI standards and in TCP/IP network management specification.

access mode A MIB access level of READ-ONLY, READ-WRITE ,WRITE-ONLY or NONE. **address translation** The process of translating a Network Layer address to a corresponding physical address for a device.

agent Software that enables a device to respond to manager requests to view or update MIB data and sends traps reporting problems or significant events.

Application Programming Interface(API) A set of routines that enable a programmer to use computer facilities. The socket programming interface and the transport layer interface are both APIs used for TCP/IP programming.

Asynchronous communication Character-oriented communication in which characters are delimited by start and stop bits.

Authentication Proof of the identity of sender of a message.

Basic Encoding Rules(BER) A set rules for translating ASN.1 values into a stream of octets to be transmitted across a network.

Berkeley Software Distribution(BSD) UNIX software which includes TCP/IP support.

Bridge A device that connects two or more physical network.

Client-server The model of interaction in a disributed system in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called client and the program satisfying the request is called a server.

Community Formally, a pair of agents with a set of application entities. Its purpose is to identify valid sources for requests and limit the scope of data that can be accessed.

Community Name Used like a password in message, validating the right of sender to access MIB data with a requested operation.

Constructed type A composite datatype, such as SEQUENCE.

Datagram The IP Protocol Data Unit.

Encapsulation The technique used by layered protocols in which a lower level protocol accepts a message from a higher level protocol and places it in the data portion of the low level frame.

Exterior Gateway Protocol(EGP) A protocol used to advertise the set of networks that can be reached within autonomous system. EGP enables this information to be shared with other autonomous systems.

Fragmentation Partitioning of a datagram into pieces. This is done when datagram is too large for network technology that must be traversed to reach the destination.

Gateway An IP router.

Group A named set of closely related MIB definitions within a module.

Internet Architecture Board(IAB) Board that oversees the Internet protocol development and standardization.

Lexicographic order Order of variables in the MIB tree, based on comparing OBJECT IDENTIFIERs from left to right until the numbers differ in a position.

Management Information Base(MIB) A logical database made up of the configuration, status and statistical information stored at a device.

Manager Software in a network management station that enables the station to send requests to view or update MIB values.

Maximum Transmission Unit(MTU) The size of the largest datagram that can be delivered across a particular path.

MIB-II A set of managed object definitions aimed managing TCP/IP based internets.

monitor A device that listens to all traffic on LAN or on a wide area link, gathering statistics, and capturing traffic that matches some specific criteria.

multihomed host A host attatched to two or more networks, and therefore requiring multiple IP addresses.

OBJECT IDENTIFIER A string of numbers derived from a global naming tree, used to identify an object.

party A network management agent or manager role. Some parties communicate without authentication, some use authentication, and some use both authentication and encryption in order to protect the privacy of data.

port number A 2-octet binary number identificatifying an upper- level user of TCP.

Primitive Type(ASN.1) A basic datatype such as an INTEGER or OCTET STRING.

Requests for Comments(RFCs) A set of documents containing Internet protocols and discussions of related topics. These documents are available online at the Network Information Center.

router A system used to connect separate LANs and WANs to an internet, and route traffic between the constituent networks.

socket The abstraction provided by Berkeley 4BSD UNIX that allows an application to access the TCP/IP protocols. It can be viewed as a pairing of an IP address and a port number.

TCP The TCP/IP standard transport level protocol that provides the reliable, full duplex, stream service on which many application protocols depend.

UDP The TCP/IP standard protocol that allows an application program on one machine to send a datagram to an application program on another machine.